N89-15566

GRAPH–BASED REAL–TIME FAULT DIAGNOSTICS

S. Padalkar, G.Karsai and J. Sztipanovits
Vanderbilt University, Nashville, TN 37235 USA

ABSTRACT

A real-time fault detection and diagnosis capability is absolutely crucial in the design of large-scale space systems. Some of the existing AI-based fault diagnostic techniques like expert systems and qualitative modelling are frequently ill-suited for this purpose. Expert systems are often inadequately structured, difficult to validate and suffer from knowledge acquisition bottlenecks. Qualitative modelling techniques sometimes generate a large number of failure source alternatives, thus hampering speedy diagnosis.

In this paper we present a graph-based technique which is well suited for real-time fault diagnosis, structured knowledge representation and acquisition and testing & validation. A Hierarchical Fault Model of the system to be diagnosed is developed. At each level of hierarchy, there exist fault propagation digraphs denoting causal relations between failure-modes of subsystems. The edges of such a digraph are weighted with fault propagation probabilities and fault propagation time intervals. Efficient and restartable graph algorithms are used for on-line speedy identification of failure source components.

INTRODUCTION

A very high degree of automation and complexity is evident in modern industrial plants and space systems. This trend towards fully automatic and largely unmanned complex systems necessitates the development of a real-time fault detection and diagnosis capability. Such a capability would lead to shorter repair times and longer system operational times, thus enhancing productivity. Signal processing techniques coupled with modern sensors are capable of fault detection and alarm generation. Advances in computer technology such as multi-processing allow improvements in real-time performance. New artificial intelligence (AI) programming techniques, such as declarative languages and symbolic processing are very efficient for representing and processing the failure models of systems.

PROBLEM STATEMENT

A real-time fault diagnostics system has to function in an environment where new alarms may constantly be generated, due to the propagation of failures. To cope with such a time-changing scenario the diagnostics system must have the following characteristics:

- Signal Processing, Alarm Generation and Failure Source Identification software must be as fast as possible. The first two are usually standard well-defined and analyzed algorithms, and hence, virtually all speed improvements have to be achieved in the failure source identification phase.
- The diagnosed results must be updated as time elapses and new alarm information is received. These results must be accurate but need not have a fine resolution. This implies that in the early stages of diagnosis a large

component such as the Gas-Delivery Assembly can be identified as the fault source. The resolution of this fault source is further refined with the passage of time and additional alarm information to a unique valve inside the Gas-Delivery Assembly.

- The User-Interface must present the current status of diagnosis in a comprehendible manner, reflecting the level and the granularity of the system under diagnosis, at which the diagnostics system is operating.

SURVEY OF OTHER TECHNIQUES

Rule-Based approaches or Expert Systems (1) have been the primary AI technique used for fault diagnostics. Expert Systems use IF - THEN rules as their knowledge representation structure, and an inference engine operating on these rules for detecting the source of failure. For diagnosing large-scale systems, Expert Systems are often unsuitable since they cannot be efficiently modularized. Large Expert Systems also suffer from maintanence, testing and validation problems. Often large Expert Systems remain incomplete because of knowledge acquisition problems.

Another approach has centered on using qualitative models (2) which are a simulation of faulty system behavior. Fault sources are identified by comparing incoming data with all possible qualitative simulation models until a match is found. This process may generate too many models and be too time-intensive for use in real-time fault diagnostics.

A variety of graph-based techniques such as fault trees (3), event-trees (4) and cause-consequence diagrams (5) have also been used for fault diagnostics. An interesting approach proposed by Narayanan and Vishwanadham combines hierarchical fault propagation digraphs with fault trees (6). It is judged that a graph-based technique offers the best hope for real-time fault diagnostics.

GRAPH-BASED APPROACH

The basic philosophy of our graph-based approach is based upon Multiple-Aspect modelling. The system under consideration is hierarchically decomposed from many aspects in order to yield many different models. A functional decomposition leads to the Hierarchical Process Model (HPM) and a structural decomposition leads to a Hierarchical Physical Component Model (HPCM). A Hierarchical Fault Model (HFM) is developed in the context of HPM with links to the HPCM.

The technique of hierarchical decomposition is widely used during model building for the following reasons:

1. Design, knowledge acquisition and knowledge-base maintanence of large complex system becomes structured and easier.
2. Running the same graph algorithms on smaller number of nodes many times takes lesser time than running them on the entire set of nodes in a system. For example it takes longer time to run an $O(n^3)$ algorithm on a graph with 200,000 nodes than it takes to run the same algorithm 200 times on a graph with 100 nodes.
3. It is possible to conduct the search for the failure source on the HFM in a parallel manner, thus enabling speedy diagnosis.
4. In most cases a large granularity component assembly can be identified as a failure source at an early stage, and then the search need only proceed in that component's part of the model.

Hierarchical Process Model

A process in the HPM can be thought of as a functional unit carrying out a specific function in the system, by utilizing different physical components. Different processes on the same level may

interact with each other through shared physical components. Processes in the HPM can be associated with many different components in the HPCM as shown in figure 1. In the context of each process the following are acquired:

1. Process Failure-Modes.
2. Process Alarms and alarm-generators. The alarm-generators accept sensor inputs and if needed, generate the appropriate alarm.
3. Alarm Failure-Mode associations.
4. Failure-Mode Physical Component associations.

Hierarchical Fault Model

Each process in the the HPM also has its fault model. This model is derived from that its failure-modes, and if present, the failure-modes of its subprocesses. All these failure-modes form nodes of a fault propagation digraph, with directed edges between individual failure-modes signifying a fault propagation possibility. Each edge in this graph is weighted with two parameters a fault propagation probability and a fault propagation time interval in terms of a minimum and a maximum. The fault propagation digraph of a process on level i is shown in figure 2. The collection of all such fault propagation digraphs and failure-mode physical components associations results in the HFM. It is possible to extract a rough fault propagation digraph from process physical interactions since most faults can only propagate along physical connections.

Model Building

The process of model building and specification was aided by graphical editing tools developed at Vanderbilt University (7). Each model was built using its its own specific editor and also had its own declarative specification language. The output of an editor is the specification of a model in its declarative language. A sample output of the fault model editor is shown in figure 3. These generated specifications are used by special-purpose interpreters for generating the run-time environment of the real-time fault diagnostic system.

DIAGNOSTIC ALGORITHMS

By running suitable algorithms on a fault propagation digraph, a failure source process and its source failure-mode can be found. Since each failure-mode in each process is also associated with physical components, the source faulty components can also be found. This process can be migrated to lower levels of process hierarchy in order to get a better resolution. Hence the failure source identification process consists of two algorithms the Failure Source Process Identification (FSPI) and the Fault Source Component Identification (FSCI). An Inter Level Migration (ILM) process does the task of searching the process hierarchy for the best resolution of the possible faulty source component.

Failure Source Process Identification

The FSPI algorithm gets as input the fault-propagation digraph of a process to be diagnosed. It also receives all alarms currently ringing within that process and its subprocesses. This algorithm is accurately capable of detecting under most circumstances, whether a single or a multiple fault occured in the process. On completion, this algorithm returns the possible fault source subprocesses and their fault source failure-modes. It uses the following constraints to determine the fault source in case of a single fault condition :

1. Reachability Constraint : All ringing alarms shall be reachable from the detected source failure-modes.
2. Monitor Constraint : No failure-mode with a normal alarm shall lie on a path from any of the detected source failure-modes to any of the failure-modes with a ringing alarm.

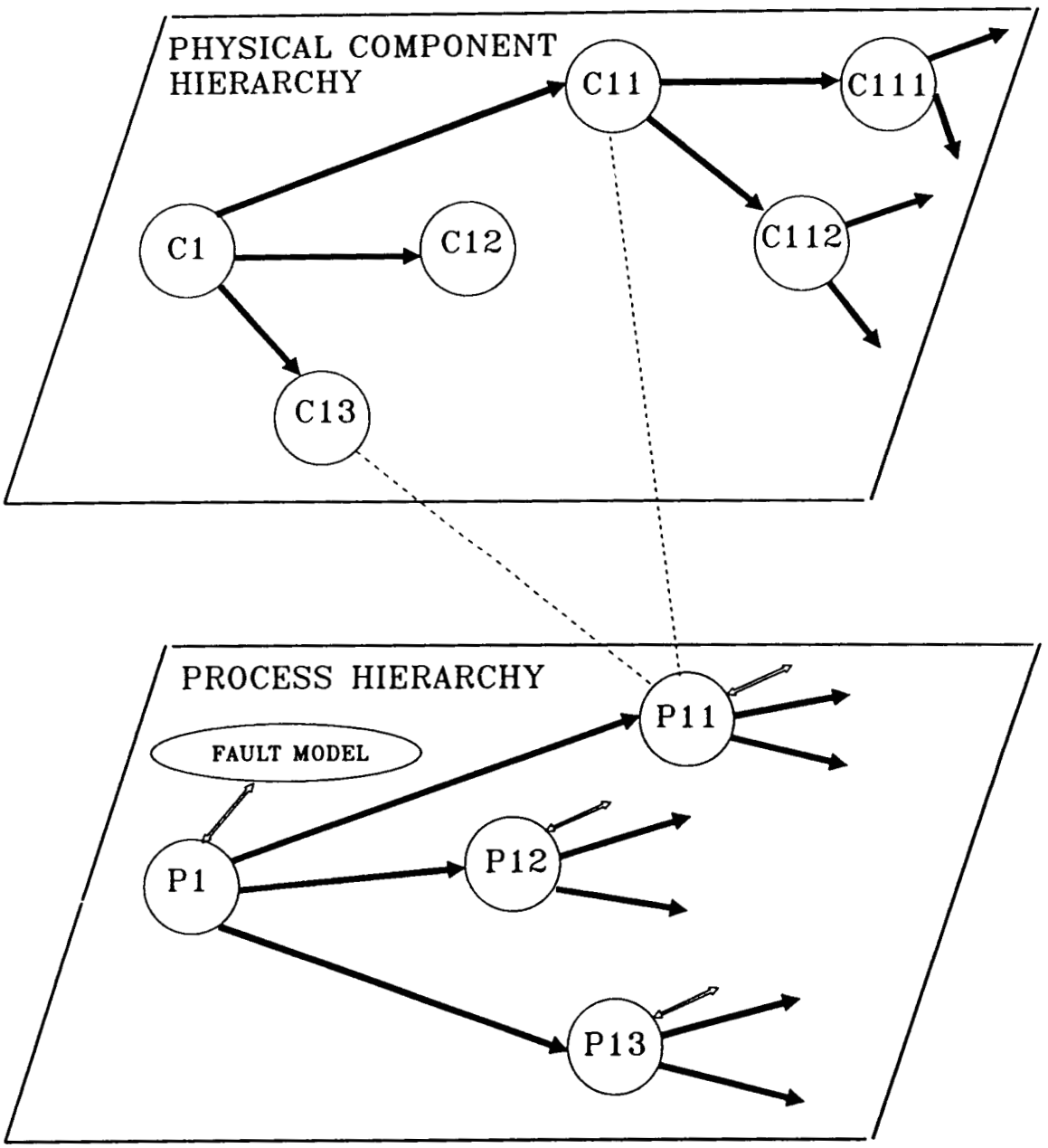
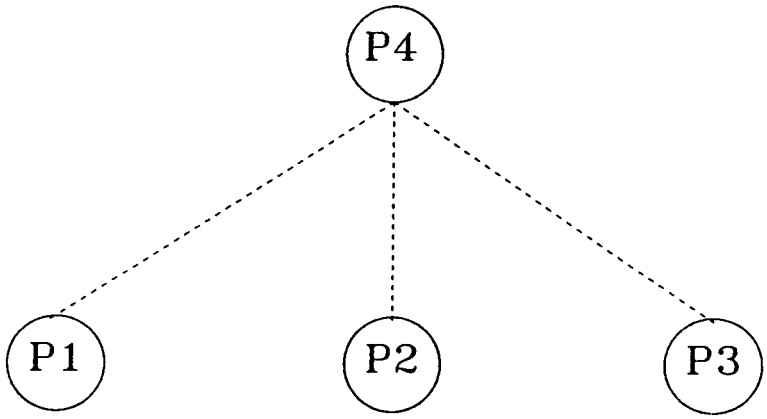


FIGURE 1 Multiple-Aspect Modelling

Process Structure On Level i :



Fault Model On Level i :

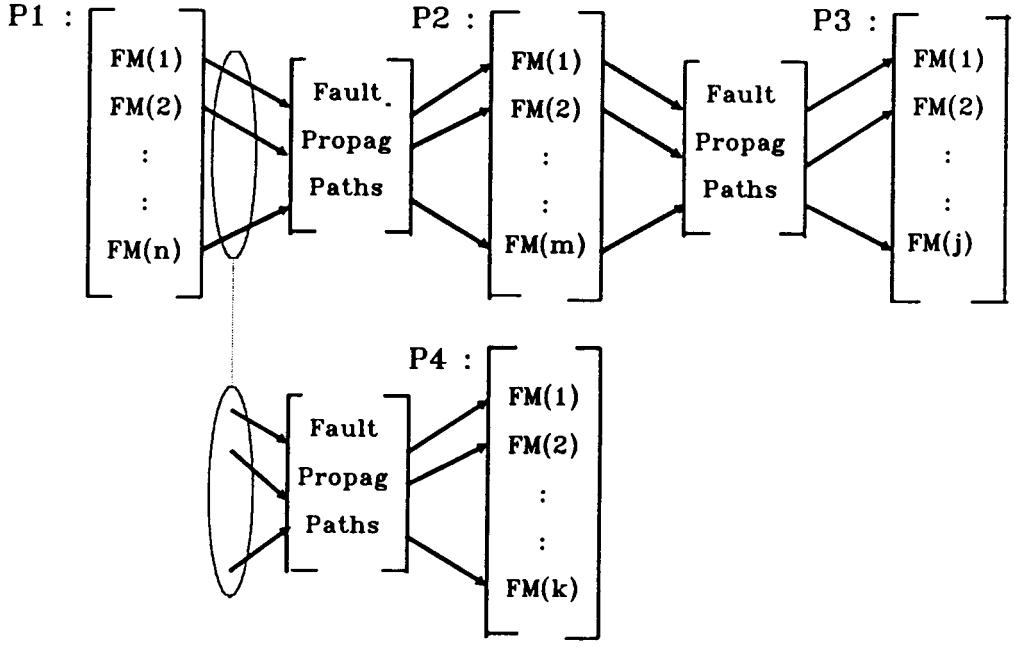


FIGURE 2: Fault Propagation Digraph

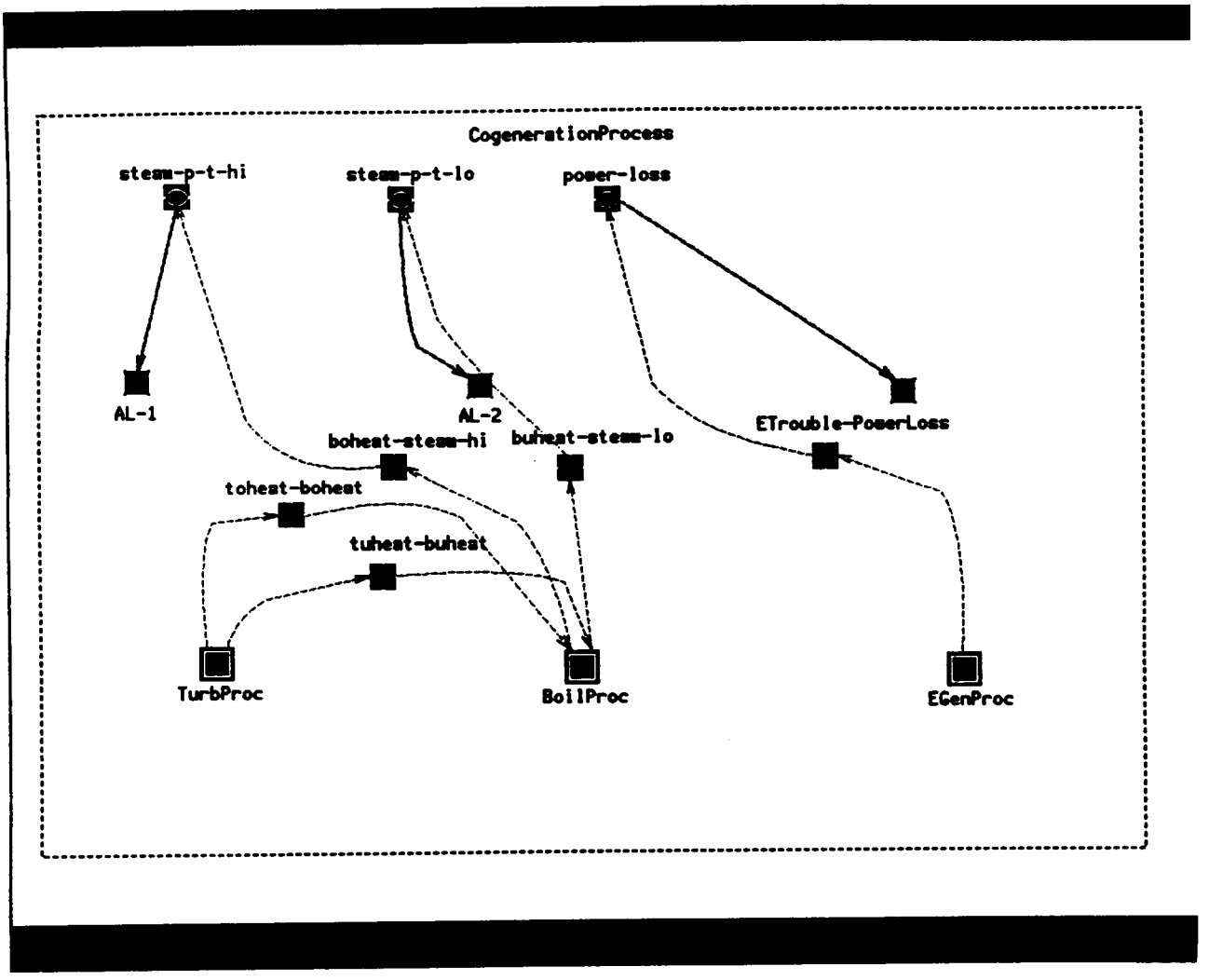


FIGURE 3: Fault Model Editor Session

3. Temporal Constraint : All ringing alarms shall be individually reachable from the detected individual source failure-modes within the time interval computed from the time intervals found on shortest path between each individual alarm and source failure-mode pair.

4. Consistency Constraint : There shall be no failure-mode with a ringing alarm whose reachability time from a source failure-mode is greater than, the maximum reachability time of a failure-mode with a normal alarm from that detected source failure-mode.

The algorithm is closed and complete and is thus suitable for speedy location of failure source processes.

Fault Source Component Identification

The FSCI algorithm takes as input a list of detected source failure processes and their source failure-modes. In case of a single fault condition it returns a union of all physical components associated with the source failure-modes. In case of a multiple fault condition it tries to find a common component amongst all the source failure-modes. If successfull it returns that common component, and if not it returns a union of all associated components.

Inter Level Migration

The ILM process detects the highest level of the process in which alarms are ringing. It then tries to search for a failure source by running the FSPI and later the FSCI algorithms on all processes in that level. The results are used to guide a breadth-first search of all processes present in the next lower level. This process continues until the lowest level of hierarchy is reached. At this point the best possible resolution of the failure source is achieved. If during this migration an alarm rings in a higher level than the current one under processing, the ILM goes to that higher level and restarts the diagnosis. At any point in time the ILM can present its best guess of the failure source in any level of process hierarchy.

DIAGNOSTIC SYSTEM ARCHITECTURE

The Real-Time Fault Diagnostic System required:

1. The use of a distributed computing architecture,

2. Support for a concurrent programming model and

3. Integration of symbolic and numerical computations.

The Multigraph Architecture (MGK) (8) has been used as a generic framework in the implementation of the diagnostic system. The MGK is dataflow oriented computing system, capable of allocating computing nodes on a distributed network consisting of uniproccessor as well as multiprocessor configurations. The language of the these computing nodes can be Lisp or C or Ada, thus enabling integration of symbolic and numerical computations. The MGK supports programming models such as autonomous communicating objects (9).

The diagnostic system architecture is shown in figure 4. A Monitor task handles the job of acquiring sensor outputs and alarm-generation. The Diagnostic task consists of a diagnostic manager object, a diagnostic methods object and a display manager object. The diagnostic manager accepts as input all generated alarms and is in charge of conducting the inter-level search for the failure source. During this search it may send a process to the diagnostic methods object asking it to perform either the FSPI algorithm or the FSCI algorithm on it. The diagnostic methods object performs the requisite algorithm and reports the result back to the diagnostic manager. These results are used by the diagnostic manager

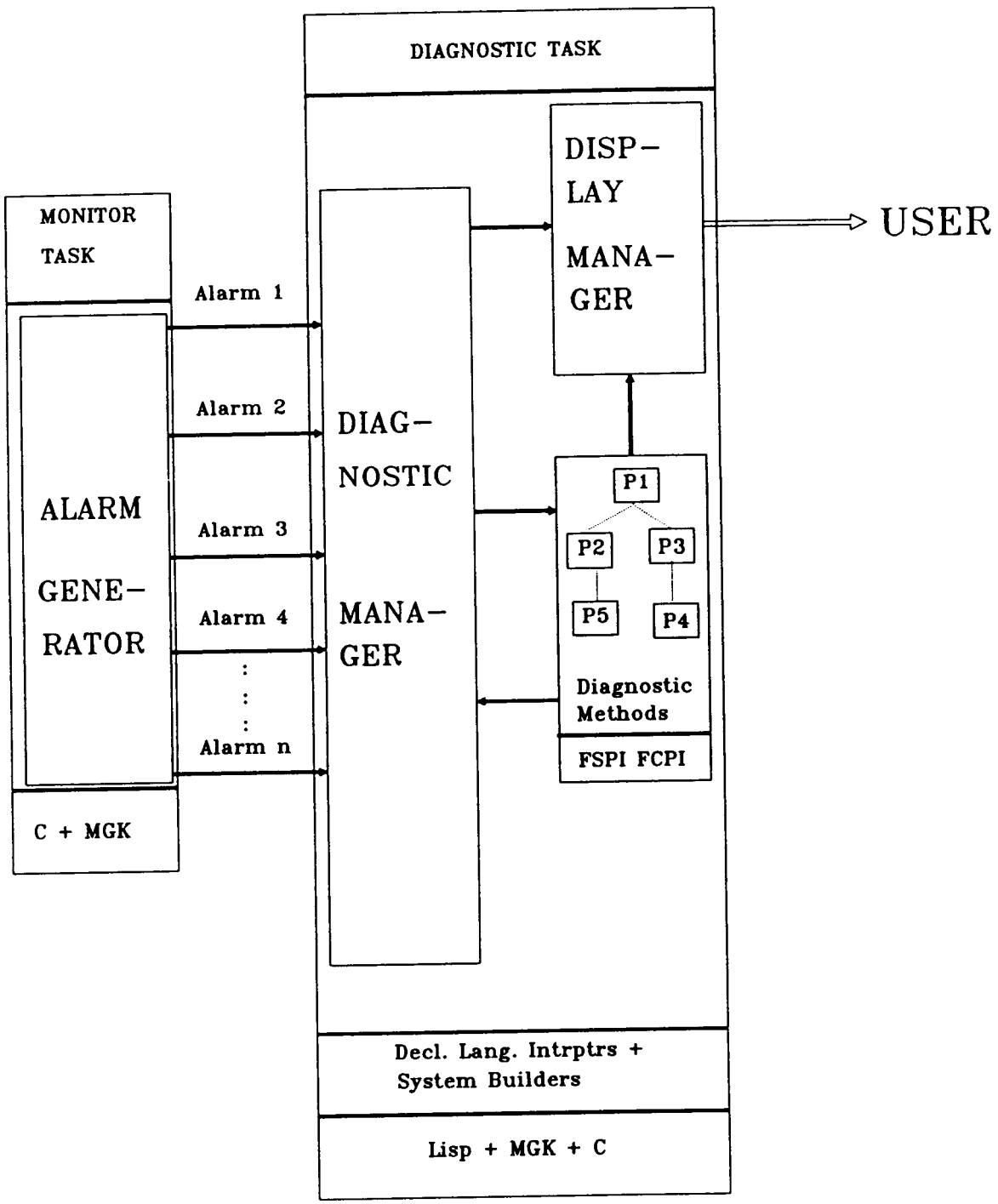


FIGURE 4: Diagnostic System Architecture

as a guide in its search. As soon as results are obtained for a level in the hierarchy they are sent over to the display manager for displaying them to the user.

TESTING AND VALIDATION

A real-time alarm pattern simulator is used to test and validate the real-time fault diagnostics. This simulator is automatically derived from the HFM. This simulator accepts as input any number of failed components and the times at which they are supposed to have failed. It then generates in real-time the pattern of alarms that would ring due to the failed components. These alarms serve as the input for the diagnostics system.

CURRENT STATUS

A real-time fault diagnostics system for a Cogenerator plant currently exists on an HP 9000/300 computer. The process model has 5 levels of hierarchy and 45 processes. The average number of failure-modes per process is about 4 and hence the average number of nodes in a fault propagation digraph is about 10. An alarm pattern simulator is currently being used to test and validate the diagnostics system. A test in which 5 alarms were generated in a span of 20 seconds, the diagnostics system located the failure source with the finest possible resolution in 25 seconds.

CONCLUSIONS

A hierarchical graph-based model appears to be the most suitable fault model for real-time fault diagnostics. The knowledge acquisition process can be automatized to a large extent. The graph algorithms developed are fast enough to be used for speedy diagnosis. The system is able to update itself and restart the diagnostic procedure if necessary. The breadth-first search strategy enables the system to provide an accurate diagnosis of a coarse resolution that can be refined with passage of time and more alarm information. Testing and validation of such a system is easier since the test program can be automatically generated from the fault model itself.

REFERENCES

(1)Laffey, T., Perkins, W., Nguyen, T., "Reasoning about Fault Diagnosis with LES", First Conference on A.I. Applications, Dec. 1984.

(2)Milne, R., "Strategies for Diagnosis", IEEE Transactions on Systems, Man, and Cybernetics, May/June 1987.

(3)McCormick, N., "Reliability and Risk Analysis", Academic Press Inc. New York, New York, 1981.

(4)Kumagai, N., Ishida, Y. and Tokumaru, H., "A Knowledge Representation for Diagnosis of Dynamical Systems", IFAC Real Time Programming, Lake Balaton, Hungary 1986.

(5)Himmelblau, D., "Fault Detection and Diagnosis in Chemical and Petrochemical Processes", Elsevier Scientific Publishing Company, Amsterdam, Netherlands, 1978.

(6)Narayanan, N. and Vishwanadham, N., "A Methodology for Knowledge Acquisition and Reasoning in Failure Analysis of Systems", IEEE Transactions on Systems, Man, and Cybernetics, March/April 1987.

(7)Karsai, G., "Graphical Description Package", unpublished memorandum, Dept. of Electrical Engineering, Vanderbilt University, 1986.

(8)Sztipanovits, J., "Execution Environment for Intelligent Real-Time Systems", Proc. of the NASA Workshop on Telerobotics, Pasadena, CA, 1987.

(9)Sztipanovits, J., Biegl, C., Karsai, G., Padalkar, S., and Purves, R., "Programming Model for Coupled Intelligent Systems in Distributed Execution Environments", Proc. of SPIE's symposium on "Advances in Intelligent Robotic Systems", Cambridge, MA, 1886.